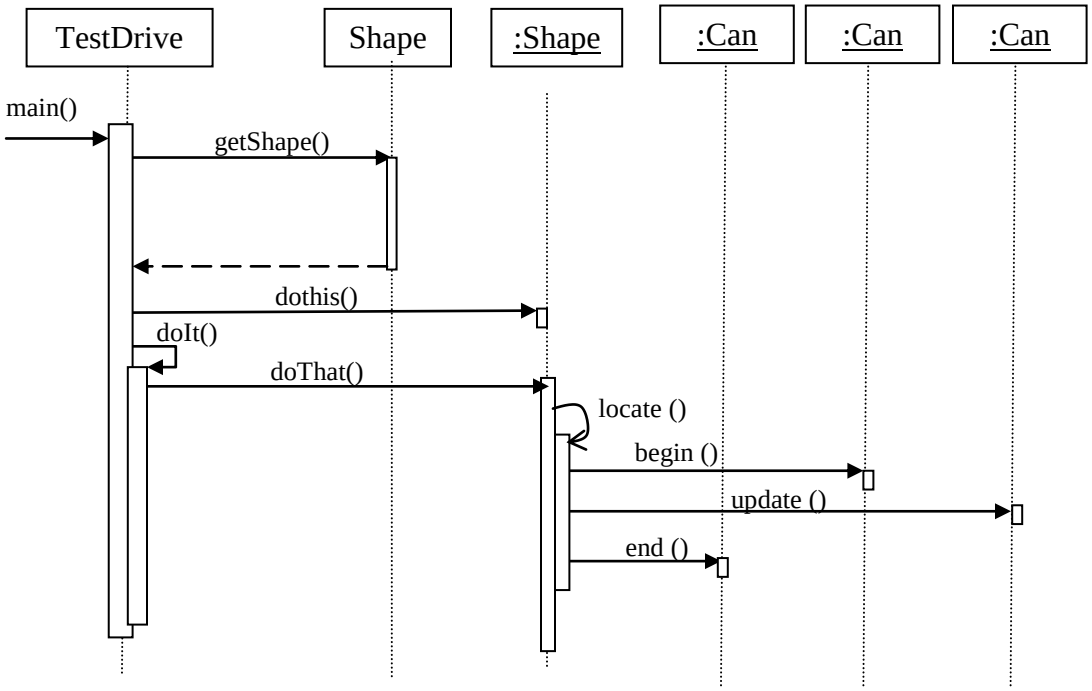
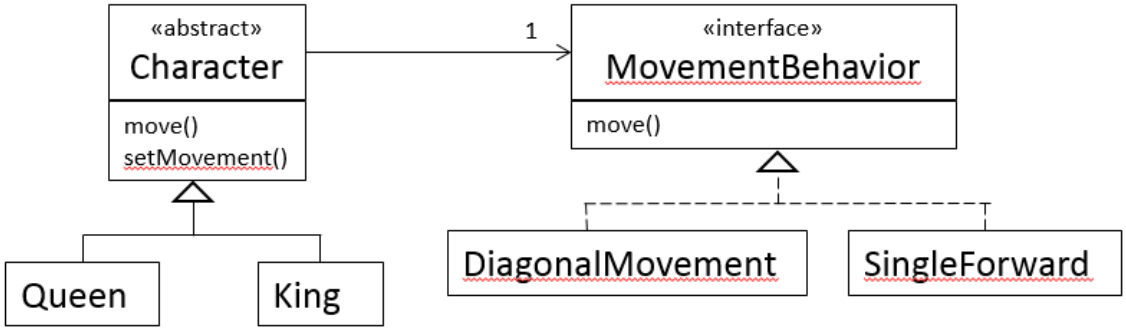
Name: _____

Student Number: _____

1. Consider the following sequence diagram and draw (to the extent possible) a class diagram showing method names for each class.



2. Consider the partial implementation of Chess in the UML class diagram below. Movement for a chess pieces is managed via the Strategy pattern. Code for these classes is given on page 6.



Consider that the following code executes at the start of the game:

1. king = new King();
2. queen = new Queen();
3. king.move();
4. queen.move();
5. queen.setMovementBehaviour(new SingleForward);
6. queen.move();

What is the object diagram at the time line 3 executes?
What is the sequence diagram for lines 4, 5, and 6?

3. Suppose we are designing an application where Investor objects need to be informed of changes in the price of a Stock object. If the price of a stock changes then the stock must inform interested investors of this change.

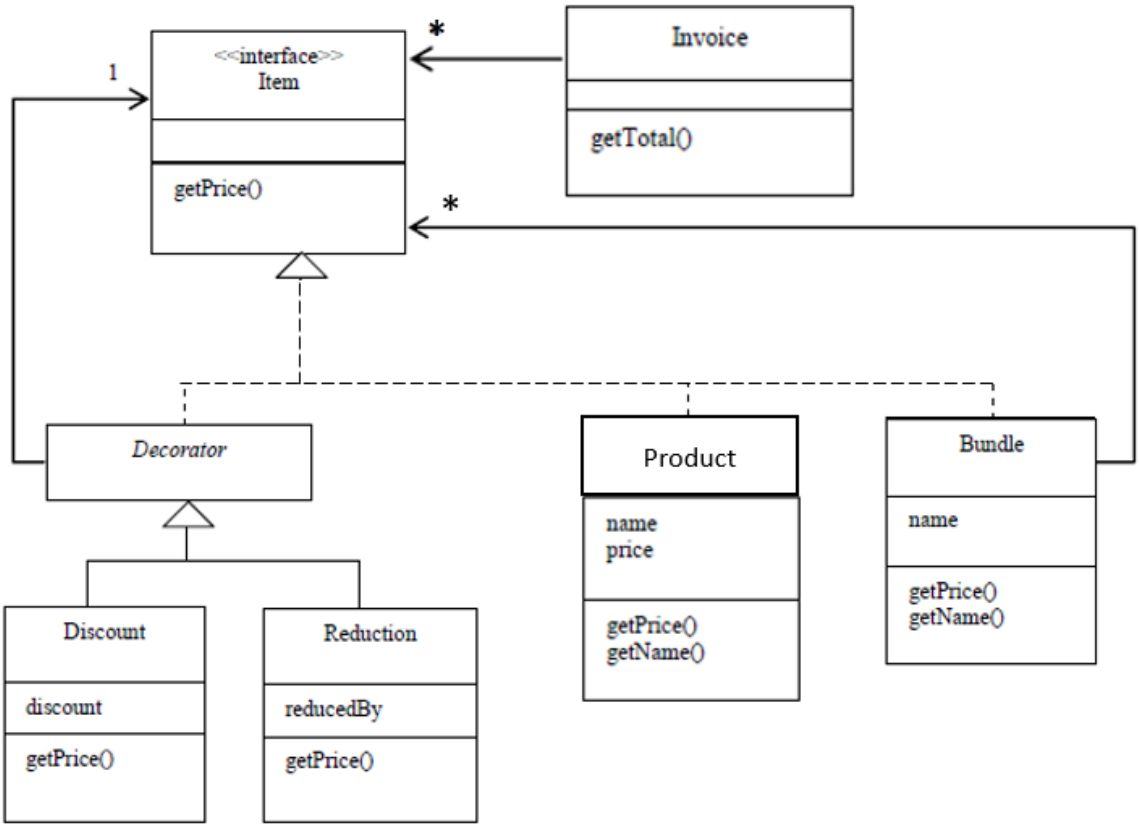
   Each Stock has a name and two prices: previous price and new price. Assume the Stock class has a method, priceChanged() that is responsible for detecting a change in the stock price. Each Investor has a name and a phone number.

   Considering the implementation must adhere to the Observer design pattern:

   a) Draw a class diagram that includes classes, interfaces, associations, multiplicities, and methods. Use reasonable names for Observer.

   b) Beginning with priceChanged() executing, draw a sequence diagram to show the messages sent when the Walmart stock must inform two observers: Steve and Amanda, that the Walmart price has changed.

4. The following class diagram is for managing invoices. Note the design involves both the Decorator pattern and the Composite pattern.



Consider an invoice with a single line item. The line item has a quantity of 3 for a bundle named "Special". This bundle comprises two products:

- a CD named O2014, with a price reduction of $10 and discount of 10%,
- a CD named O2013 (not discounted or reduced).

The bundle is discounted by 20%.

Use a sequence diagram to show all messages sent when the invoice receives the getTotal() message and determines its total.

Pseudocode for the pertinent methods is on page 5.

Pseudocode for Invoices, Products, Reductions, Discounts, Bundles

```
getPrice() in Discount is:
        return  item.getPrice() * (1-(discount/100))


getPrice() in Reduction is:
        return  item.getPrice() - reducedBy


getPrice() in Product is:
        return  price * quantity


getPrice() in Bundle is:
          total = 0
        for each item in bundle:
            total = total + item.getPrice()
        return  total


getTotal()  in Invoice is:
        total = 0
        for each item:
                total = total + item.getPrice()
        return total
```

Chess Code

```
public class Character {
   private MovementBehaviour movementBehaviour;
   String move() {
     return movementBehaviour.move();
   }
   public void setMovementBehaviour(MovementBehaviour movementBehaviour) {
      this.movementBehaviour = movementBehaviour;
   }
}

public class King extends Character {
   public King() {
      setMovementBehaviour(new SingleForward());
   }
}

public class Queen extends Character {
   public Queen() {
      setMovementBehaviour(new DiagonalMovement());
   }
}

public interface MovementBehaviour {
   String move();
}

public class SingleForward implements MovementBehaviour {
   public String move() {
     return "move one step forward";
   }
}

public class DiagonalMovement implements MovementBehaviour {
   public String move() {
      return "Moving Diagonally";
   }
}
```